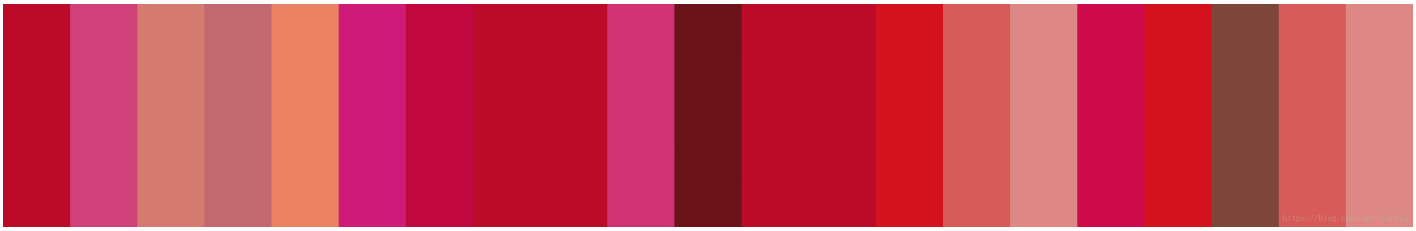
BugkuCTF-MISC多彩 writeup

转载

carlzj 于 2018-09-18 10:40:15 发布
分类专栏： CTF 文章标签： CTF BUGKU 多彩 MISC YSL


CTF 专栏收录该内容
1 篇文章 0 订阅
订阅专栏

这可真是一道脑洞大开的让人骂X的题呢。。。



找了好久终于找到了答案，大家还是自己看吧。

https://fadec0d3.blogspot.com/2018/03/n1ctf-2018-lipstick.html

既然评论区有小伙伴表示访问不了，那我就直接复制过来吧，但是我真的懒得翻译了，大家自己看吧~

This challenge was very interesting! It included multiple types of stego and some web scraping.

We're given the image shown above. Looking very innocuous with flat colors, it seems one would try to hide lsb data in it. Let's try zsteg to extract some values:

```
$ zsteg -a 603fda27-2893-4e60-9d9f-39962b8c8440.png

b1,bgr,lsb,xy       .. text: "flag.txt"
b2,g,lsb,xy         .. file: 5View capture file
b2,g,msb,xy         .. file: VISX image file
b2,b,msb,xy         .. text: "UUuUUWUUUUUuUUUU"
b2,bgr,lsb,xy       .. text: "AAAP@QUDQ"
b3,b,lsb,xy         .. file: GLS_BINARY_MSB_FIRST
b4,r,lsb,xy         .. text: "UDTTUEEDDEDEUEDDTEDETDEDDEDDDDTDUEEETDUDEUDDDDDDDDDDDDUDDDDDDDDDDDDDDDDDDDD\"
b4,r,msb,xy         .. text: "333333333;"
b4,g,lsb,xy         .. file: 5View capture file
b4,g,msb,xy         .. file: VISX image file
b4,b,msb,xy         .. text: ["w" repeated 12 times]
b6,g,msb,xy         .. text: "4ES4EQ4MS4M"
```

Looks like one of the first hits is promising! "flag.txt", maybe we should extract that portion alone, first let's inspect it.

Passing the -v flag into zsteg, we can get the hex dump of that extraction:

```
zsteg -v 603fda27-2893-4e60-9d9f-39962b8c8440.png b1,bgr,lsb,xy
b1,bgr,lsb,xy       .. text: "flag.txt"
    00000000: 00 00 00 00 00 00 00 e3  50 4b 03 04 14 00 01 08  |........PK......|
    00000010: 08 00 70 42 67 4c 37 20  ff 48 4d 00 00 00 43 00  |..pBgL7 .HM...C.|
    00000020: 00 00 08 00 00 00 66 6c  61 67 2e 74 78 74 ce 98  |......flag.txt..|
    00000030: 5e 65 7a ba 27 91 a6 eb  1b 25 54 f3 b9 6d 0f ca  |^ez.'....%T..m..|
    00000040: 78 7c 20 6a 79 e8 39 99  c8 df ad 3d 9a c8 4b fc  |x| jy.9....=..K.|
    00000050: 53 1d db 6e 95 9d 07 ae  1c 91 eb fe a7 18 33 00  |S..n..........3.|
    00000060: 9e f5 12 8e 7f c6 e6 7e  1a c8 3d cf 94 97 2f b0  |.......~..=.../.|
    00000070: 96 64 f3 a7 d1 94 64 23  98 3f 47 50 4b 01 02 3f  |.d....d#.?GPK..?|
    00000080: 00 14 00 01 08 08 00 70  42 67 4c 37 20 ff 48 4d  |.......pBgL7 .HM|
    00000090: 00 00 00 43 00 00 00 08  00 24 00 00 00 00 00 00  |...C.....$......|
    000000a0: 00 20 00 00 00 00 00 00  00 66 6c 61 67 2e 74 78  |. .......flag.tx|
    000000b0: 74 0a 00 20 00 00 00 00  00 01 00 18 00 5b d9 a1  |t.. .........[..|
    000000c0: f6 a9 b5 d3 01 a5 f5 f6  76 a9 b5 d3 01 a5 f5 f6  |........v.......|
    000000d0: 76 a9 b5 d3 01 50 4b 05  06 00 00 00 00 01 00 01  |v....PK.........|
    000000e0: 00 5a 00 00 00 73 00 00  00 00 00 92 49 24 92 49  |.Z...s......I$.I|
    000000f0: 24 92 49 24 92 49 24 92  49 24 92 49 24 92 49 24  |$.I$.I$.I$.I$.I$|
```

On the top there's PK, that's the magic byte for zip! A zip hidden in lsb.

Extracting it with zsteg we have some trailing bytes, we can extract the zip itself with binwalk and verify using 7z:

```
$ zsteg 603fda27-2893-4e60-9d9f-39962b8c8440.png -E b1,bgr,lsb,xy > out
$ binwalk -e out
$ cd _out.extracted
$ 7z l 8.zip

   Date      Time    Attr         Size   Compressed  Name
------------------- ----- ------------ ------------  ------------------------
2018-03-06 17:19:31 ....A           67           77  flag.txt
------------------- ----- ------------ ------------  ------------------------
2018-03-06 17:19:31                 67           77  1 files
```

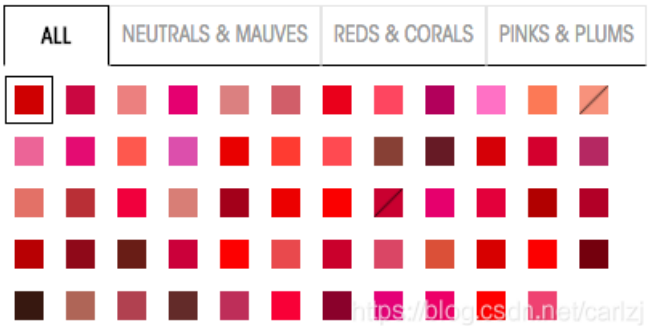
Next we needed to get the password. At this point we could assume they decided to go with a value from rockyou.txt, but asking the admins it turned out there was another part to the challenge missing.

Reframing with the idea of the challenge named "Lipstick" it seemed to suggest more than just lsb. There are 21 colors, which may map to some character for the zip, but we would need to map the colors to numbers somehow. It is also important for the stego designer to make this somewhat deterministic.

Looking through the color channels again in StegSolve, this turns up:

YSL: Yves Saint Laurent. A recognizable fashion brand which sells lipstick. This may lead us to something predictable!

Looking for lipstick on their online store we can find the full color palette - example.



The other interesting part here is that these colors have clear class names and distinct background colors. This is perfect for identifying colors in the challenge!

Here's a small script which was used to scrape the colors from the palette:

```
a = {};

for (let i=0; i<59; ++i) {
  a[$('.swatch_color')[i].style['background-color']] = $('.swatch_color')[i].title;
}

JSON.stringify(a, null, 2);
```

This gives us a convenient JavaScript object we may query:

```
{
  "rgb(188, 11, 40)": "1 Le Rouge - Blood Red (Satin)",
  "rgb(184, 52, 75)": "04 Rouge Vermillon - Raspberry Red (Satin)",
  "rgb(221, 136, 133)": "06 Rose Bergamasque - Delicate Nude Pink (Satin)",
  "rgb(207, 26, 119)": "7 Le Fuchsia - Pure Saturated Fuschia (Satin)",
  "rgb(206, 135, 133)": "10 Beige Tribute - Dark Nude (Satin)",
  "rgb(194, 105, 111)": "11 Rose Carnation - Soft Peony Rose (Satin)",
  "rgb(213, 29, 55)": "13 Le Orange - Bright Orange (Satin)",
  "rgb(234, 94, 107)": "17 Rose Dahlia - Coral Orange (Satin)",
  "rgb(161, 24, 96)": "19 Fuchsia Pink - Bright Fuschia Pink (Satin)",
  "rgb(238, 133, 199)": "22 Pink Celebration - Bubble Gum Pink (Satin)",
  "rgb(235, 130, 98)": "23 Coral Poetique - Pink Coral (Satin)",
  "rgb(233, 152, 131)": "24 Blond Ingenu - Frosted Coral (Satin)",
  "rgb(218, 116, 155)": "26 Rose Libertin - Bright Purple Pink (Satin)",
  "rgb(208, 65, 121)": "27 Fuchsia Innocent - Hot Pink (Satin)",
  "rgb(235, 105, 92)": "36 Corail Legende - Orange Coral (Satin)",
  "rgb(202, 102, 174)": "49 Tropical Pink - Hot Pink (Satin)",
  "rgb(212, 18, 29)": "50 Rouge Neon - Bright Red (Satin)",
  "rgb(235, 85, 71)": "51 Corail Urbain - Medium Peach Orange (Satin)",
  "rgb(246, 98, 96)": "52 Rosy Coral - Rosy Coral (Satin)",
  "rgb(126, 69, 58)": "53 Beige Promenade - Browd Red (Satin)",
  "rgb(94, 35, 41)": "54 Prune Avenue - Dark Maroon (Satin)",
  "rgb(193, 9, 43)": "56 Orange Indie - Mauve (Satin)",
  "rgb(192, 8, 62)": "57 Luminous Pink - Magenta (Satin)",
  "rgb(165, 63, 103)": "58 Luminous Mauve - Bubblegum Pink (Satin)",
  "rgb(212, 122, 111)": "59 Golden Melon - Golden Orange (Satin)",
  "rgb(170, 64, 64)": "66 Rosewood - Soft Pink Nude (Satin)",
  "rgb(220, 53, 77)": "67 Rouge Heroine - Gold Metallic (Satin)",
  "rgb(203, 132, 123)": "70 Le Nu - Basic Nude (Satin)",
  "rgb(148, 23, 41)": "72 Rouge Vinyle - Deep Raspberry (Satin)",
  "rgb(215, 11, 22)": "73 Rhythm Red - Magenta Pink (Satin)",
  "rgb(229, 35, 23)": "74 Orange Electro - Electric Orange (Satin)",
  "rgb(183, 48, 62)": "75 Rose Mix - Deep Rose (Satin)",
  "rgb(209, 50, 116)": "76 Explicit Pink - Spring Pink (Satin)",
  "rgb(206, 10, 74)": "77 Fuschia Live - Blush Rose (Satin)",
  "rgb(161, 0, 6)": "201 Orange Imagine - Orange Red (Matte)",
  "rgb(161, 0, 52)": "202 Rose Crazy - Cranberry Red (Matte)",
  "rgb(167, 11, 35)": "203 Rouge Rock - Rich Red (Matte)",
  "rgb(131, 35, 37)": "204 Rouge Scandal - Maroon Red (Matte)",
  "rgb(97, 37, 29)": "206 Grenat Satisfaction - Brown Red (Matte)",
  "rgb(184, 20, 70)": "208 Fuchsia Fetiche - Raspberry Red (Matte)",
  "rgb(231, 51, 37)": "213 Orange Seventies - Orange with Brown undertones (Matte)",
  "rgb(215, 91, 89)": "214 Wood On Fire - Pinky Nude (Matte)",
  "rgb(184, 38, 59)": "216 Red Clash - Mauvey Red (Matte)",
  "rgb(201, 88, 108)": "217 Nude Trouble - Mauvey Beige (Matte)",
  "rgb(203, 93, 70)": "218 Coral Remix - Coral Nude (Matte)",
  "rgb(195, 1, 9)": "219 Rouge Tatouage – Vibrant Pink Red (Matte)",
  "rgb(230, 26, 1)": "220 Crazy Tangerine – Electric Orange (Matte)",
  "rgb(106, 19, 25)": "222 Black Red Code – Rust Red (Matte)",
  "rgb(51, 26, 19)": "225 - Minimal Black - Matte Finish - Black (Matte)",
  "rgb(165, 106, 92)": "340 Golden Copper - Reddish Pink (Satin)",
  "rgb(163, 77, 86)": "09 Rose Stiletto - Rich Berry Rose (Satin)",
  "rgb(92, 47, 44)": "205 Prune Virgin - Burgundy (Matte)",
  "rgb(174, 67, 95)": "207 Rose Perfecto - Blush Pink (Matte)",
  "rgb(226, 0, 74)": "211 Decadent Pink - Bright Pink (Matte)",
  "rgb(126, 32, 50)": "212 Alternative Plum - Burgundy Wine (Matte)",
  "rgb(207, 44, 125)": "215 Lust For Pink - Purple Bubblegum Pink (Matte)",
  "rgb(214, 0, 112)": "221 Rose Ink – Deep Mauve Pink (Matte)",
  "rgb(232, 11, 44)": "223 Corail Anti-Mainstream – Neon Coral (Matte)",
```

```
  "rgb(219, 90, 121)": "224 Rose Illicite – Creamy Dusty Pink (Matte)"
}
```

We can now query this based on values found inside the image. But how do we get those values in the image? At first trying macOS's digital color picker didn't work too well, instead let's use Python!

```
from PIL import Image

im = Image.open('603fda27-2893-4e60-9d9f-39962b8c8440.png')
rgb_im = im.convert('RGB')

for x in range(0, 21):
  r, g, b = rgb_im.getpixel((50 + (150 * x), 1))
  print(r, g, b)
```

This yields:

```
(188, 11, 40)
(208, 65, 121)
(212, 122, 111)
(194, 105, 111)
(235, 130, 98)
(207, 26, 119)
(192, 8, 62)
(188, 11, 40)
(188, 11, 40)
(209, 50, 116)
(106, 19, 25)
(188, 11, 40)
(188, 11, 40)
(212, 18, 29)
(215, 91, 89)
(221, 136, 133)
(206, 10, 74)
(212, 18, 29)
(126, 69, 58)
(215, 91, 89)
(221, 136, 133)
```

Now we can grep for each color in the JSON result, and produce integer values for each color's associated edition number:

```
$ IFS=$'\n'; for i in `cat image-colors` ; do grep $i ./lipsticks.json >> found; done
$ cat found | sed 's/.*: "//g;s/ .*//;s/^0*//'
1
27
59
11
23
7
57
1
1
76
222
1
1
50
214
6
77
50
53
214
6
$ cat found | sed 's/.*: "//g;s/ .*//;s/^0*//' | tr -d '\n'
1275911237571176222115021467750532146
```

Attempting this number didn't work for the password. Another hint came out about using binary, let's try with that:

```
$ cat found | cut -d: -f1 | xargs python -c 'import sys; print "".join([bin(int(x)).lstrip("0b") for x in s
101111110101111100101110001010100010110001101010010101011100100011100101010110101110
0
```

Still doesn't work. Maybe we need to convert this to the character representation:

```
$ cat found | sed 's/.*: "//g;s/ .*//;s/^0*//' | xargs python -c 'import sys; print "".join([bin(int(x)).ls
白学家
```

Then extracting the archive we scraped earlier using this password, we get the flag:

```
$ cat flag.txt
flag{White_Album_is_Really_worth_watching_on_White_Valentine's_Day}
```